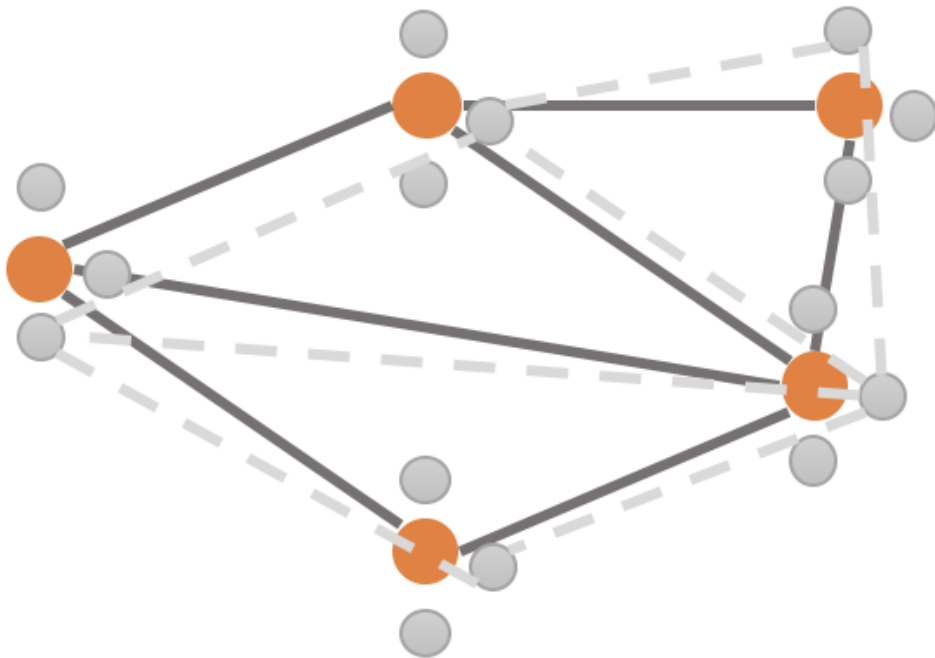


2018/01/15-2018/01/21周报

DONE

- **投稿工作**：阅读了struct2vec这篇文章，了解了后台的基本思路。附录附上我对STRUCT2VEC的理解。STRUCT2VEC主要可以将每个节点在的拓扑结构转换成一个高维向量，进而可以计算得到每个节点在拓扑结构上的相似节点。接下去，根据当前浏览器中的子图结构 S ，如何推算其相似的子图结构 S' ，就可以转换成从 S 的节点的所有相似节点中，构造出 S' 的过程，如下图，橙色点和黑色实线构成了已有的子图 S ，灰色点则是橙色点的STRUCT2VEC相似点，然后我们需要找到如图虚线所示的拓扑结构，使得和原有的拓扑结构足够相似。



- **硕士论文**：目前已完成：
 - 相关工作（图可视化布局、图可视化相关技术、大图可视化、聚类技术）
 - 基于图的关系可视分析方法（大规模图可视化方法，基本布局算法描述）
 - 可视分析系统实现（可视分析系统架构、事件机制实现、边捆绑技术）
 - 剩余的绪论和总结以及未来工作等待开元跟如晟完成即可。
- **REACT学习**：已制定新的技术栈，学习了一点react的基础知识。
- **复习《虚拟现实》**：周末在复习课程，准备下周二的考试。
- **华为的回复**：在东明基础上帮忙增加了一点点微小的工作。

TODO

- **新投稿**：下周会开始思考可视化的设计以及如何挑选子图 S' 的问题
- **虚拟现实课程考试**：准备虚拟现实课程的考试
- **硕士论文**：等待开元跟如晟的完成。

任务	截止日期	当前进度
大图可视化调研		开始调研芯片
关于palantir软件注册撰写		已和律师联系清楚，具体家东在负责
硕士论文	春节前	基本已完成
新投稿思路确定	12月底	用图嵌入的方法STRUCT2VEC的方法来进行
华为回复	22日	已完成

STRUC2VEC

原文：struc2vec: Learning Node Representations from Structural Identity

捕捉节点在网络中的结构信息，将它表达成一个高维向量，需要考虑以下两个相关的性质：

1. 不同节点的表达之间的距离（也就是高维向量之间的距离）应该跟他们的结构之间的相似度高度相关。
2. 节点的这种结构表达，不应该依赖于节点或者边的属性以及它们的标签。节点的结构信息应该跟他们在网络中的位置不相关。

STRUC2VEC的基本步骤：

1. 在不同的邻域大小下，比较图中的每个节点之间的结构相似度。
产生一个结构相似度度量的多层模型（hierarchy）。
2. 构造一个带权的多层图，网络中所有的节点都会出现在每一层，每一层都会和度量结构相似度的多层模型（hierarchy）的每一级相对应。然后带权多层图的边的权重和该边相关的节点对之间的距离成反比。
3. 使用上述的多层图生成每个节点的上下文。用带偏的random walk来生成多层图的节点序列。
4. 使用某种技术，通过节点序列给出的上下文来学习潜在表达

第一步：度量结构相似度

- $R_k(u)$ 表示了节点 u 的 k 级邻域。
- $s(S)$ 则表示节点的集合 $S \subset V$ 的度数序列。
- $g(D_1, D_2)$ 度量了两个度数序列 D_1, D_2 之间的距离
- $f_k(u, v)$ 表示了 u, v 两节点之间， k 级邻域（距离小于等于 k 的节点和所有它们之间的边）的结构距离。

$$f_k(u, v) = f_{k-1}(u, v) + g(s(R_k(u)), s(R_k(v)))$$

$$f_{-1} = 0$$

上述定义表达的是一个递归的过程，把问题转换成了如何定义两个节点集合之间的距离，也就是 $g(\cdot)$ 该如何计算。

- 文章使用了Dynamic Time Warping(DTW)来度量两个度数序列之间的距离。

序列A和B，其中， $a \in A$ ， $b \in B$ ，那么 a, b 之间的距离定义为： $d(a, b) = \frac{\max(a, b)}{\min(a, b)} - 1$ ，当 $a = b$ 时，他们的距离也就是0。然后两个序列之间的距离，也就是所有 a, b 组合之间的距离和。

其他方案也可以用在该框架下，感觉这里的时间复杂度相对较高，可以有替换的方案。VIGOR论文中提到的方法？

第二步：构造上下文图（context graph）

- 构造一个带权的多层图 M 。层数由0到 k^* ，其中 k^* 表示的是图的直径，也就是原始图 $G = (V, E)$ 中节点之间的最远距离。
- 第 k 层图表达的是节点之间 k 级邻域的相似度度量。

- 故而 M 的每一层都是一个完全图，都会有 $n = |V|$ 个节点，边的数量是 $n(n - 1)$ ，边的权重的定义：

$$w_k(u, v) = e^{-f_k(u, v)}, k = 0, \dots, k^*$$

当然，只有 f_k 定义了，边才能被定义。

- 因为 $f_k(u, v)$ 最小是0，那么边的权重 $w_k(u, v)$ 的取值范围是 $(0, 1]$ ，当等于1时，代表两个节点结构相同。
- 不同层级之间，通过有向边连接，在 k 层的每个节点都会跟 $k - 1$ 层和 $k + 1$ 层的相同节点用有向边连接，边的权重定义：

$$w(u_k, u_{k+1}) = \log(\Gamma_k(u) + e), k = 0, \dots, k^* - 1$$

$$w(u_k, u_{k-1}) = 1, k = 1, \dots, k^*$$

- 其中 $\Gamma_k(u)$ 代表了指向节点 u 的权重大于 k 层图平均权重的边的数量：
 $\Gamma_k(u) = \sum_{v \in V} \mathbf{1}(w_k(u, v) > \overline{w_k})$ ，也就是度量了该节点 u 和第 k 层其他节点之间的相似度。
- 而 $\overline{w_k} = \sum_{(u, v) \in \binom{V}{2}} w_k(u, v) / \binom{n}{2}$

第三步：生成节点的上下文

利用 M 生成节点的上下文的过程：带偏的随机游走（biased random walk）。

- 首先需要决定，随机游走的这一步，是留在当前层还是换层
- 有一定概率 q ，留在当前层，假设留在当前层，那么从节点 u 走到 v 的概率（权重越大，概率越高，也就是更容易走到相似节点上）：

$$p_k(u, v) = \frac{e^{-f_k(u, v)}}{Z_k(u)}, Z_k(u) = \sum_{v \in V, v \neq u} e^{-f_k(u, v)}$$

- 假设在另一半概率上 $1 - q$ ，换层，换到 $k + 1$ 的概率定义：

$$p_k(u_k, u_{k+1}) = \frac{w(u_k, u_{k+1})}{(u_k, u_{k+1}) + (u_k, u_{k-1})}$$

而换到 $k - 1$ 层的概率：

$$p_k(u_k, u_{k-1}) = 1 - p_k(u_k, u_{k+1})$$

- 当节点走到 $k + 1$ 层时，产生的context可能只是前 k 层产生的context的一个子集，所以这种情况下，并不没有贡献
- 每次随机游走从第0层开始，游走步数都是一个固定的，较小的值，然后随机游走会在同一个节点上重复多次，产生多个独立的游走

第四步：学习语言模型

通过word embedding的方法，对第三步生成的随机游走结果，生成高维向量。

文章使用了Skip-Gram¹。

1. Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013). [↗](#)